Remarks/Arguments

The Examiner is thanked for the careful review of this Application. Claims 1-20 are pending after entry of the present Amendment. Amendments were made to the specification, drawing, and claims to correct typographical error and better define the claimed invention. The amendments do not introduce new subject matter.

Objections to Drawings:

In the Office Action, the Office has rejected Figure 4 of the formal drawings (as received by the USPTO on January 9, 2004 and filed by the Applicants on November 20, 2001) for having a typographical error. The Applicants have corrected the typographical error in the word "PRogram" of Figure 4. Thus, the Applicants respectfully request that objection to Figure 4 be withdrawn.

Rejections under 35 U.S.C. § 101:

The Office has rejected claims 1-20 as being nonstatutory. Per Office's request and to better define the claimed invention, independent claims 1, 8, and 16 have been amended to specifically define a method for use in a computing system for processing Java Application code, a method for use in a computing system for transforming lines of a Java template for preprocessing, and a Java preprocessor for use in a computing system, respectively. As such, the Applicants respectfully submit that, amended independent claims 1, 8, and 16, and thus respective dependent claims are statutory. Accordingly, the Applicants respectfully request that rejection of claims 1-20 under 35 U.S.C. section 101 be withdrawn.

Term Interpretations:

The Office has interpreted certain terms in the specification. As described in more detail below, the Applicants respectfully submit that the Office's interpretations of the following terms are different than the definition of the terms, as provided by the Applicants in the specification of the subject application, as follows:

A. Meta language:

The Office has interpreted that the meta language refers to the Java programming language. The Applicants respectfully disagree with the Office's interpretation. One of ordinary skill in the art knows that meta languages are used to describe another language (e.g., here used in preprocessor). *See* Alan Freedman, The Computer Desktop Encyclopedia, 562 (2nd ed. 1999). While in the subject application the meta language can use the Java programming language, the meta language, as described in the subject application, should not be interpreted to be equivalent and/or limited to the Java programming language. By way of example, the meta language of the claimed invention can be C++, C, or any other selected language. As such, the Office's interpretation of the meta language is not the same as the Applicants' interpretation.

B. Header file:

This term has been interpreted by the Office to be "a file containing a PROLOG (i.e., a documents' area where programmers document the programs' authors and history), and in short a file containing comments. The Applicants respectfully submit that the Office's interpretation of the header file is not the same as the Applicants. First, in the specification of the subject application, header code is used and not header file. See page 10, lines 16-23. Second, the header code, referred to as the "prolog" in the claimed invention, can include some copyright information, Java language 'package' and 'import' clauses, and intermediate program class declaration with an opening brace '{'. Such definition, however, is not equivalent to "a file containing comments."

C. Footer file:

This term has been interpreted by the Office to be a file containing source code such as class files (e.g., the header file in C++). The Applicants respectfully submit that the Office's interpretation of the footer file is not the same as the Applicants'. Again, in the specification of the subject application, footer code or footer data is used and not footer file. See page 10, lines 16-23 and page 11, lines 21-23. Furthermore, the footer code or data is not the same as the header file in C++. Rather, as provided on page 18, lines 12-13 of the subject application, the footer data can include the definition of the 'main' method and a closing brace '}'.

D. Intermediate program:

This term is interpreted to be the output from the preprocessor ready to be used as input to the Java compiler. The Applicants respectfully submit that the Office's interpretation of the term intermediate program is not the same as the Applicants'. In the subject application, the Java object text file is to be compiled by the compiler so as to generate a Java byte-codes file. See page 10, lines 5-6. In the subject application, contrary to the Office's interpretation, the intermediate program is generated as a result of the Java object text file being processed by the first preprocessing stage. Thereafter, the "intermediate program" is compiled by the Java compiler so as to generate the executable intermediate program. At this point, the executable intermediate program is executed generating the "intermediate source." Then, the intermediate source is compiled by the Java compiler so as to generate the executable code. In the prior art preprocessors such as C/C++, however, the source file is preprocessed directly generating the "intermediate source" which is then compiled by the C/C++ compiler so as to generate the executable code. Accordingly, while the C/C++ compiler compiles the "intermediate source" so as to arrive at the executable code, in the subject application, the executable intermediate program of the subject application is executed so as to create the intermediate source.

F. Meta code:

This term has been interpreted by the Office to be the preprocessor file that is an input to the preprocessor that includes directives that may or may not have arguments and Java code. The Applicants respectfully disagree with the Office's interpretation. On page 2, lines 13-14,

the Applicants state that meta code is lines of text in a source file, which start with a particular symbol. On page 8, lines 4-6, the Applicants state that the Java template text file comprises a Java language program and meta code for use with the Java preprocessor. Thus, contrary to the Office's interpretation, meta code cannot constitute the entire preprocessor file input. Rather, the preprocessor file input includes meta code (i.e., lines of meta code) as well as the source code.

Additionally, on page 12, lines 2-8, the Applicants provide:

> In addition to processing normal object text, the Java preprocessor of the present invention if further capable of processing string mode text. String mode text is text that does not begin with a meta symbol and occupies multiple lines in a file. More particularly, the line of text includes LineFeed or end-of-line symbols. As described in greater detail subsequently, the preprocessor of the embodiments of the present invention is capable of operating on strings that occupy multiple lines as though the string comprised a single line of text.

Thus, contrary to the Office's interpretation, the meta code of the claimed invention can be any construct of Java, and may or may not include a directive with or without arguments.

G. Meta symbol:

This term has been interpreted to be the actual indicator that a line is meant to be processed by the preprocessor (i.e., formally called a directive.). The Applicants herein submit that a line of the meta code can be required to be processed by the preprocessor even if the line of meta code does not include a directive.

Rejections under 35 U.S.C. § 103(a):

The Office has rejected claims 1-20 under 35 U.S.C. 103(a) as being anticipated by Java the programming language as taught by the text book "JAVA Primer Plus" by Paul M. Tyma et al., as published on March 6, 1996 (herein after the Java Reference) in view of the text book "The Annotated C++ Reference Manual" (ANSI Base Document) written by Margaret Ellis and Bjarne Stroustrup, as published on June 7, 1990 (herein after the C++ Reference). Specifically, the Office interprets that "...it would have been obvious to one of ordinary skill in the art at the time of the invention to add a preprocessor like the preprocessor in C++ to the JAVA programming language because a preprocessor adds the capability for a programming language to perform '...macro substitution, conditional compilation, and inclusion of named files.'"

For at least the reasons provided below, none of the combinations of the cited prior art raise a *prima facie* case of obviousness against the subject matter defined in independent claims 1, 8, and 16 because as a whole and in its entirety, the Java reference leads away from using a preprocessor in the Java language. It is respectfully submitted that the Office has dismissed the portions of the Java Reference leading away from implementing a preprocessor with the Java language. The Office's attention is drawn to the following excerpts:

> Java has also done away with the standard #define macros. Actually, they've done away with the C preprocessor altogether. This is continuing with the "enforced OOP ideology. Java doesn't use header files #defines when compiling source code. Say good-bye to those huge header files listing function prototypes, typedefs, defines. etc.
>
> This is a little strange for those of us who are accustomed to putting all of our constants in header files. But actually it works out fine. The idea is that an object is instanced with the necessary constants defined. What would normally be a constant that was declared with a #define is now made constant with the keyword final. Any data type that is preceded with final is not changeable once it is defined. This also helps enforce stronger type checking.....
>
> ...
>
> Function macros can easily be translated to classes or methods within the classes that would use them. A class, interface, or method can be created just as easily as a macro, and can actually be more useful in certain circumstances defined as a method rather than as a macro. [Emphasis added.]

Thus, taking the Java Reference as a whole and in its entirety, specifically the underlined statements emphasized above, one of ordinary skill in the art would not have been motivated to incorporate the same components (e.g., macros and preprocessor) deemed unnecessary by the Java Reference. Additionally, one of ordinary skill in the art notes that Java can do without such features due to Java's platform independence.

Furthermore, the Applicants respectfully submit that all the combined elements and limitations of independent claims 1, 8, and 16 have not been considered as a whole by the Office. For instance, the Office has not demonstrated a prior art teaching or suggestion of the intermediate program, compiling of the intermediate program to create and intermediate class, and generating of an object text file from the intermediate class (as defined in claim 1); transforming of a line text into a function argument when a line of text in the source code does not begin with a meta symbol (as defined in independent claim 8); and a meta code converter capable of converting a Java template to create an intermediate program, and an object generator that compiles the intermediate program to create the intermediate class, and thereafter an object text file (as defined in independent claim 16).

Still further, the claims have not been given their broadest reasonable interpretation. It is well established that the claims should be interpreted in view of the specification, and the pending claims must be given their broadest reasonable interpretation consistent with the specification. In particular, when the specification provides definitions for terms appearing in the claims, the claims should not be read in vacuum, and the specification can be used in interpreting claim language. Thus, the Applicants respectfully submit that the Office's interpretation of the claims do not reflect the interpretation consistent with the specification in view of the Office's unclear and vague interpretations of the terms, as described in detail above with respect to rejection of the claims under section 101.

Yet further, the combinations of the cited references fail to disclose, teach, or suggest all the claim limitations as provided in the claimed invention. By way of example, among

other features, the Java preprocessor has the capability to preprocess both, statements, which have directives, and statements, which do not have directives. This is contrary to the teachings of the C/C++ preprocessors wherein preprocessing is limited to only statements having directives. In fact, even if the Java Reference and the C++ Reference were combinable and were to be combined (a preposition with which the Applicants disagree), the combination of the two references would not have disclosed, suggested, or taught preprocessing statements not have directives.

Still further, in the claimed invention, the meta language can use an entire programming language, e.g., Java. In the C++ or C preprocessors, the meta language has not been taught, disclosed, or suggested to be capable of using an entire programming language. Furthermore, in the claimed invention, the meta language can use any programming language, and is not limited to the Java programming language, only. The C or C++ preprocessor, however, do not have such capabilities.

Accordingly, independent claims 1, 8, and 16 are respectfully submitted to be patentable under 35 U.S.C. § 103(a) over any combination of the cited prior art. In a like manner, dependent claims 2-7, 9-15, and 17-20 each of which directly or indirectly depends from the applicable independent claim are submitted to be patentable under 35 U.S.C. § 103(a) over any combination of the cited prior art for at least the reasons set forth above regarding the independent claims 1, 8, and 16.

The Applicants respectfully request examination on the merits of the subject application, and respectfully submit that all of the pending claims are in condition for allowance. Accordingly, a notice of allowance is respectfully requested. If the Examiner has any questions concerning the present Amendment, the Examiner is kindly requested to contact the undersigned at (408) 774-6913. If any additional fees are due in connection with filing this Amendment, the Commissioner is also authorized to charge Deposit Account No. 50-0805 (Order No. SUNMP014). A duplicate copy of the transmittal is enclosed for this purpose.

Respectfully submitted,
MARTINE PENILLA & GENCARELLA , LLP

Fariba Yadegar-Bandari, Esq.
Reg. No. 53,805

710 Lakeway Drive, Suite 170
Sunnyvale, CA 94085
Telephone (408) 774-6913
Facsimile (408) 749-6901
Customer No. 32291

Amendments to the Drawings:

The attached sheet of drawings includes changes to FIG. 4. This sheet replaces the original sheet including FIG. 4.

Attachment: Replacement Sheet